



EE 466/586
VLSI Design


Partha Pande
School of EECS
Washington State University
pande@eecs.wsu.edu




Two-Phase Systems

Lecture 20


Clocking disciplines

- 
- Rules for constructing sequential machines.
 - Combinations of registers and gates.
 - Behavior of clocks and primary inputs over time.
 - Rules are sufficient to guarantee that the system will work at some clock rate.
 - May not be as fast as we want.

Clocking Rules

- 
- Combinational logic gates cannot be connected in a cycle
 - All components must have bounded delay

Flip-flop rules

- 
- Primary inputs change after clock (ϕ) edge.
 - Primary inputs must stabilize before next clock edge.
 - Rules allow changes to propagate through combinational logic for next cycle.
 - Flip-flop outputs hold current-state values for next-state computation.

Signals in flip-flop system



positive clock edge




ϕ




S



Latch-based machines

- 
- Latches do not cut combinational logic when clock is active.
 - Latch-based machines must use multiple ranks of latches.
 - Multiple ranks require multiple phases of clock.

Two-sided latch constraint

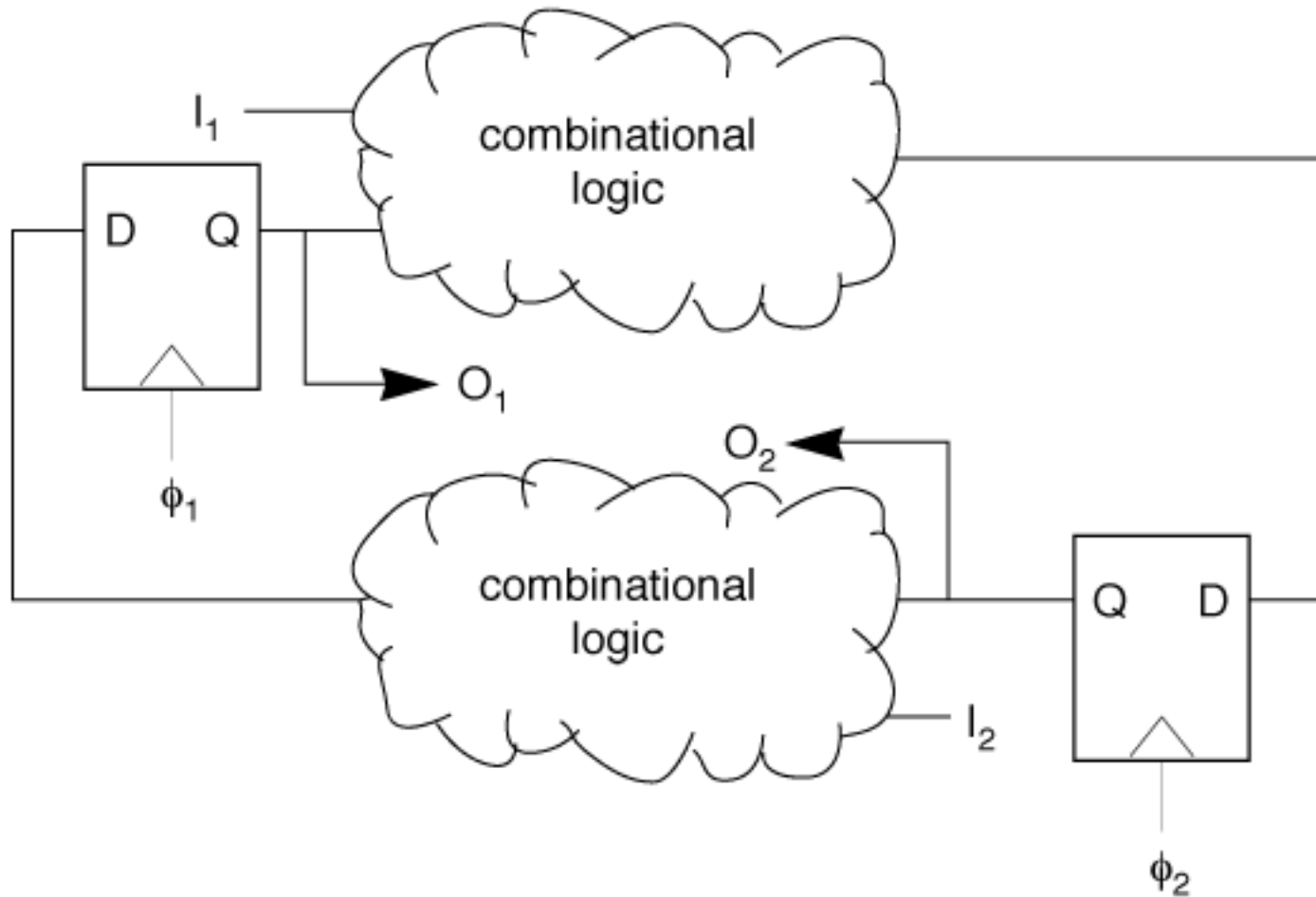
- 
- Latch must be open less than the shortest combinational delay.
 - Period between latching operations must be longer than the longest combinational delay.
 - Note: difference between shortest and longest combinational delay may be large (sum_0 vs. sum_{31}).

Strict two-phase clocking discipline



- Strict two-phase discipline is conservative but works.
- Can be relaxed later with proper knowledge of constraints.
- Strict two-phase machine makes latch-based machine behave more like flip-flop design, but requires multiple phases.

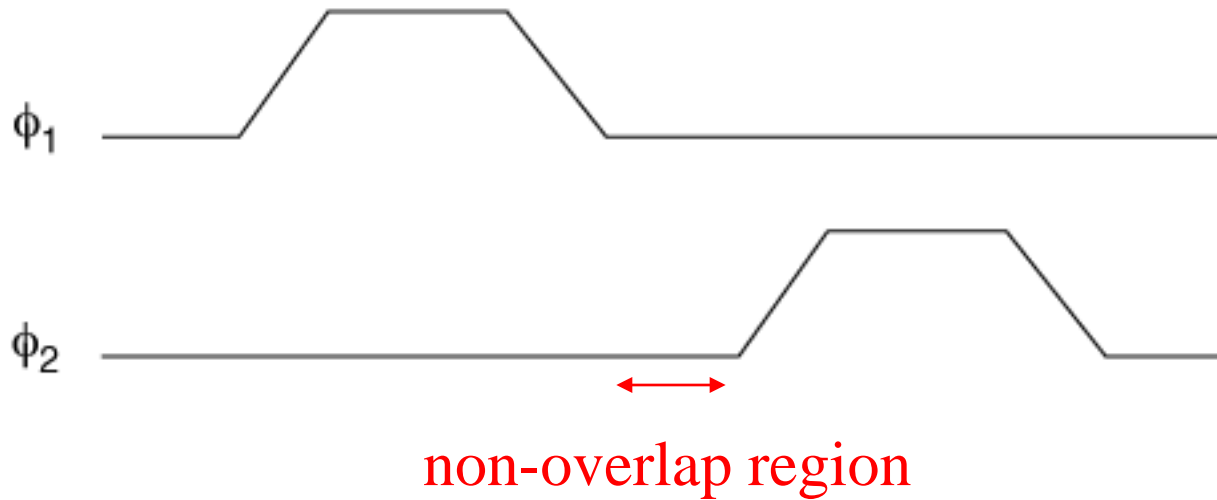
Strict two-phase architecture




Two-phase clock




Phases must not overlap:




Why it works

- 
- Each phase has a **one-sided** constraint: phase must be long enough for all combinational delays.
 - If there are no combinational loops, phases can always be stretched to make that section of the machine work.
 - Total clock period depends on sum of phase periods.


Clocking types

- 
- Logic on different phases operate at different times—can't mix signals from different phases.
 - Primary inputs must obey the same rules as internal signals.
 - Clocking types help us to ensure that machine structure is valid.

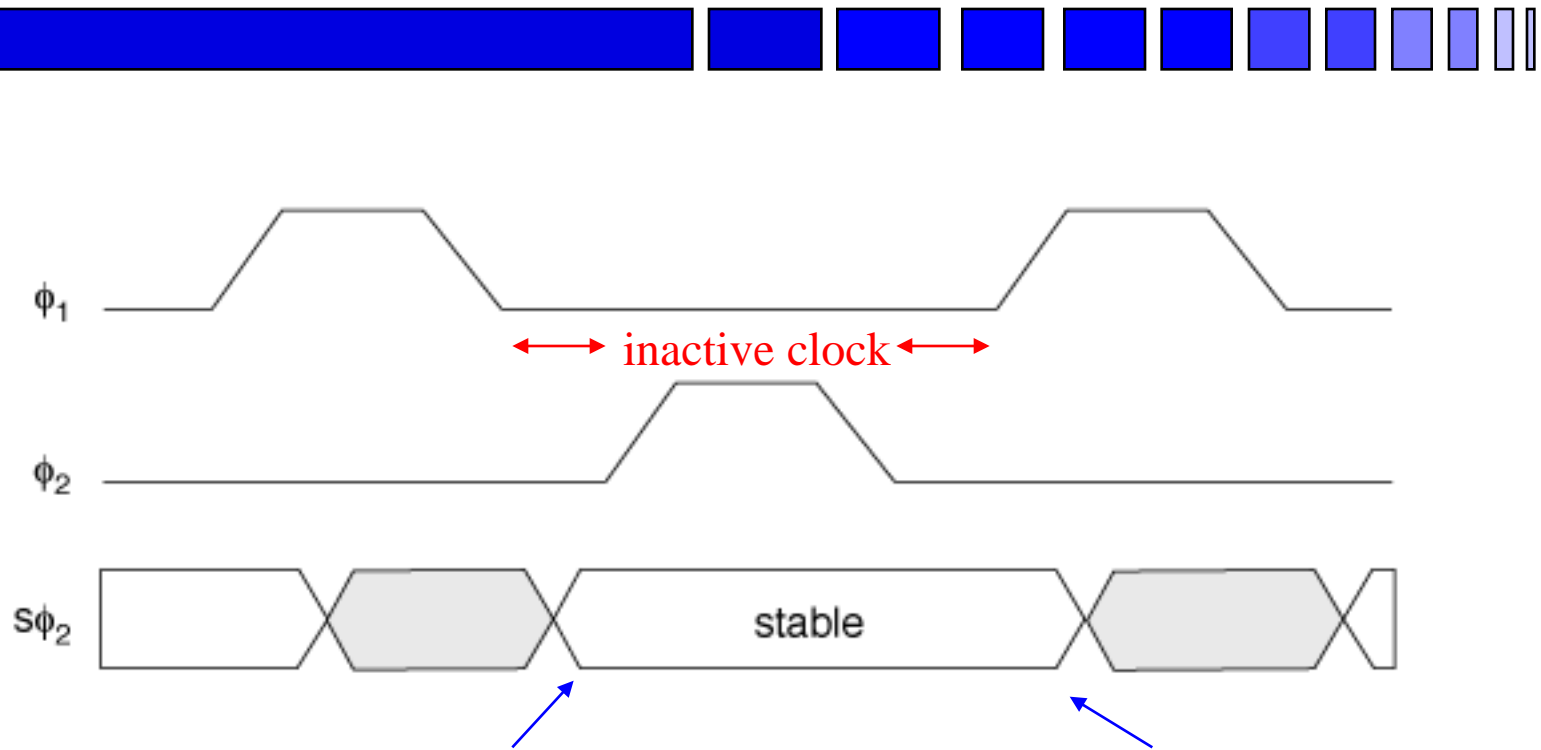
Stable signals

- 
- A logic signal is always stable during one phase—phase in which the latch which produced it is not active.
 - Easiest to think of machine behavior in terms of stable signals, though signals propagate while not stable.

Signal types

- 
- Clocks are separate type: ϕ_1 , ϕ_2 .
 - Two types of stable data signal:
 - stable ϕ_1 (s ϕ_1)
 - stable ϕ_2 (s ϕ_2)
 - A stable signal has a complementary valid signal:
 - stable ϕ_2 (s ϕ_2) = valid ϕ_1 (v ϕ_1)

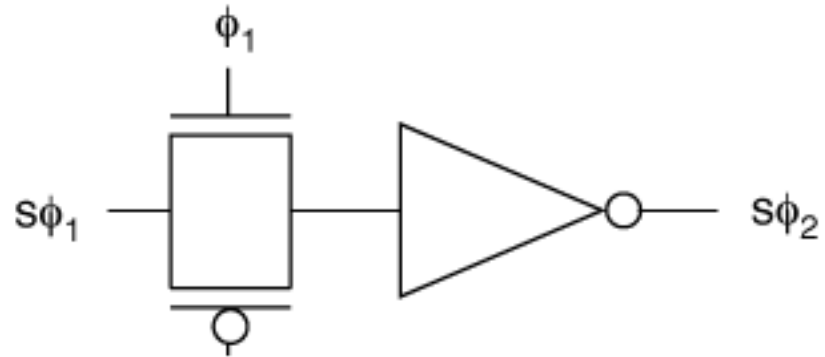
Stable data signal



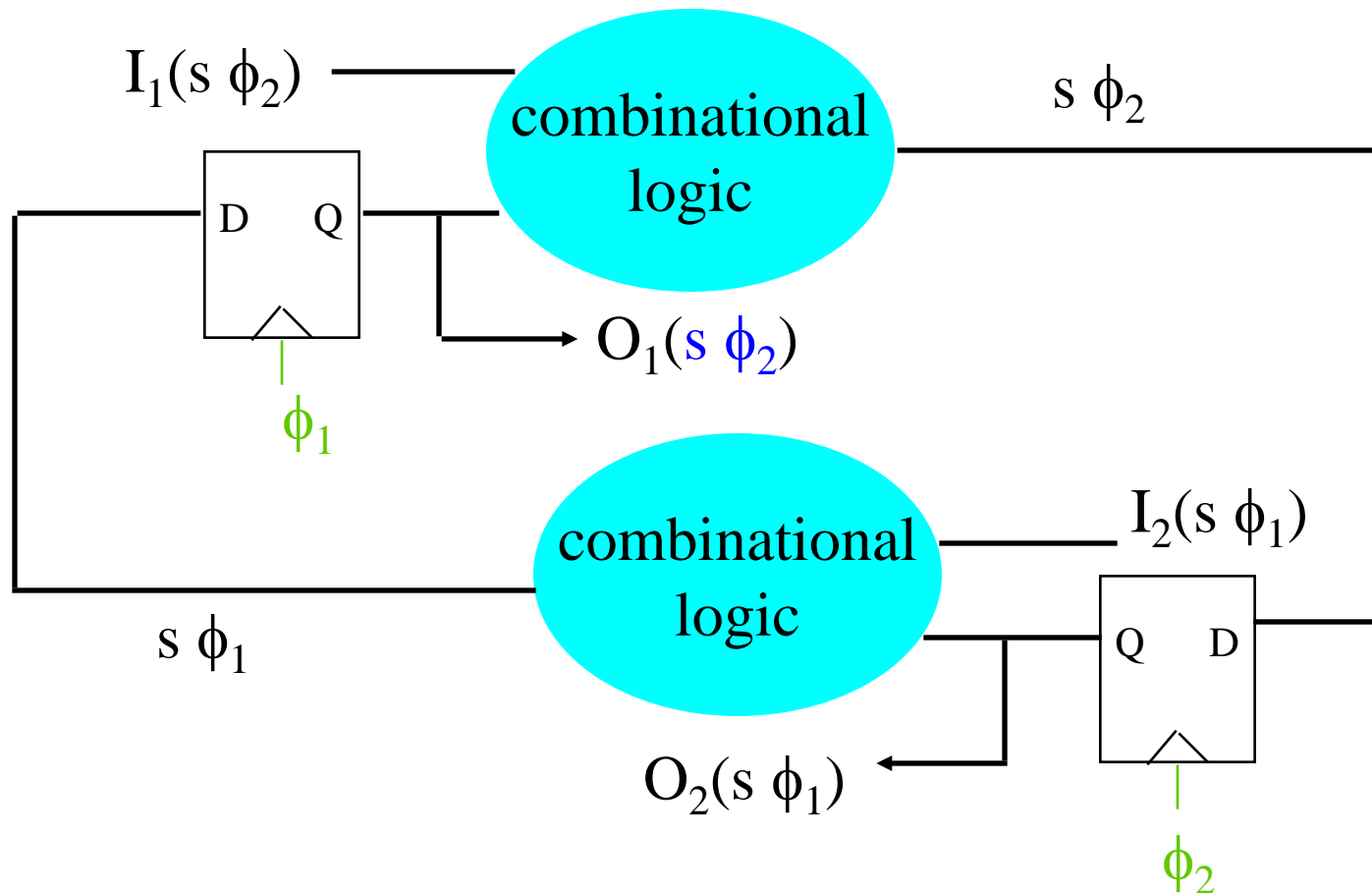
stable ϕ_2 becomes
valid at end of ϕ_1

stable until latch
feeding this
logic goes active

How clocking types combine

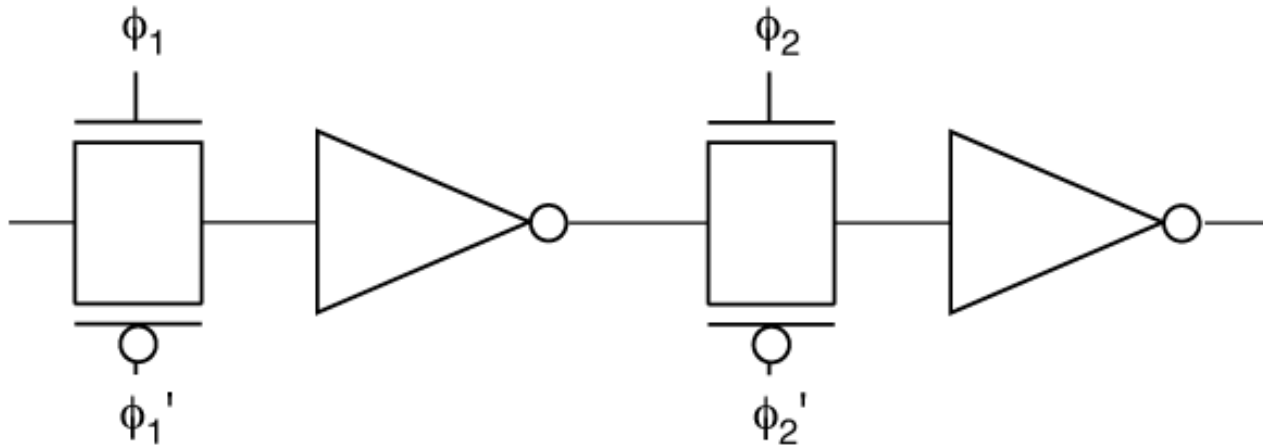


Clocking types in the two-phase machine

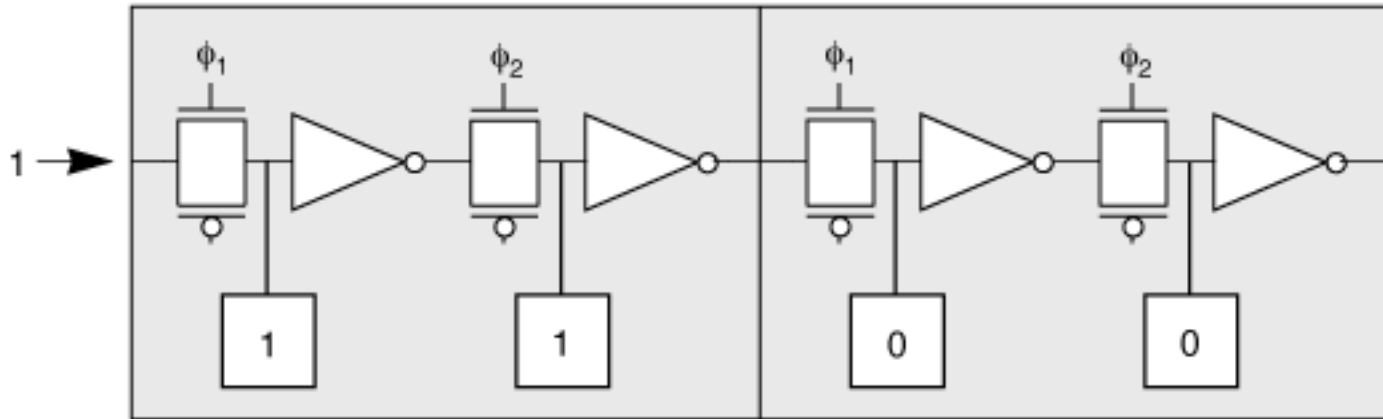


Example: shift register

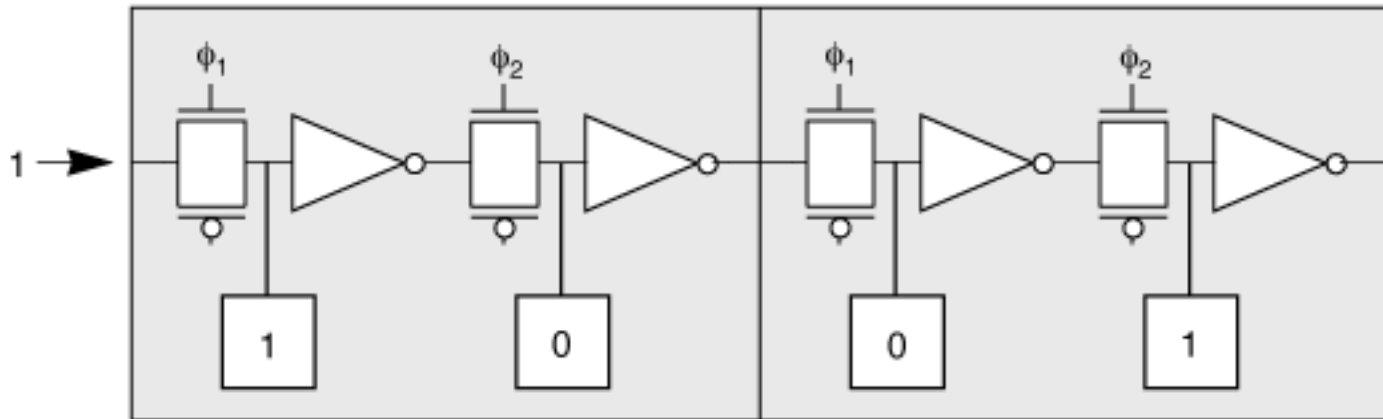
- Want to displace bit by n registers in n cycles.
- Each register requires two phases:



Shift register operation




$$\phi_1 = 1, \phi_2 = 0$$




$$\phi_1 = 0, \phi_2 = 1$$

Non-strict disciplines

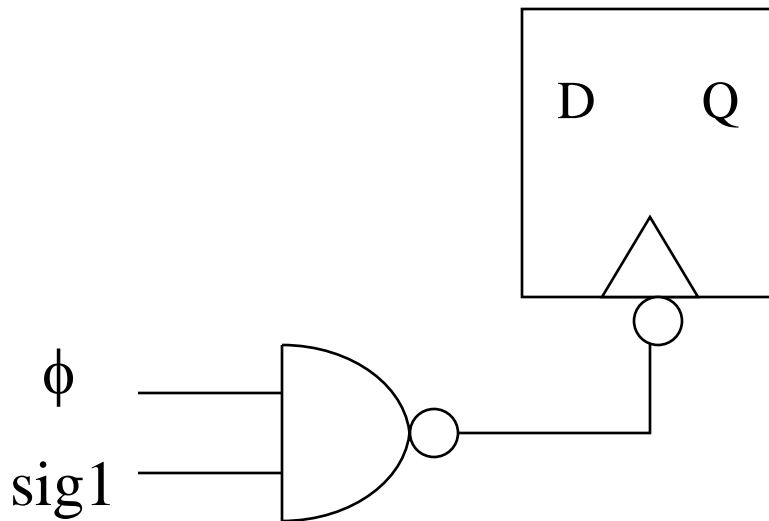
- 
- Some relaxation of the rules can be useful:
 - reduce area;
 - increase performance.
 - Rules must be relaxed in a way that ensures the machine will still work.

Qualified clocks


- 
- Use logic to generate a clock signal which is not always active.
 - Qualification must not introduce glitches into the clock—glitches violate the fundamental definition of a clock by introducing extra edges.
 - Use stable signals to qualify clocks.

Qualified clock


- Clock logically combined with signal:



Uses of qualified clocks

- 
- May want to conditionally load a register.
 - May qualify a clock to turn off machine for low-power operation.
 - Latch must not lose its value during inactive period.
 - Difficult to ensure that logic value will come high in time—use quasi-static latch.

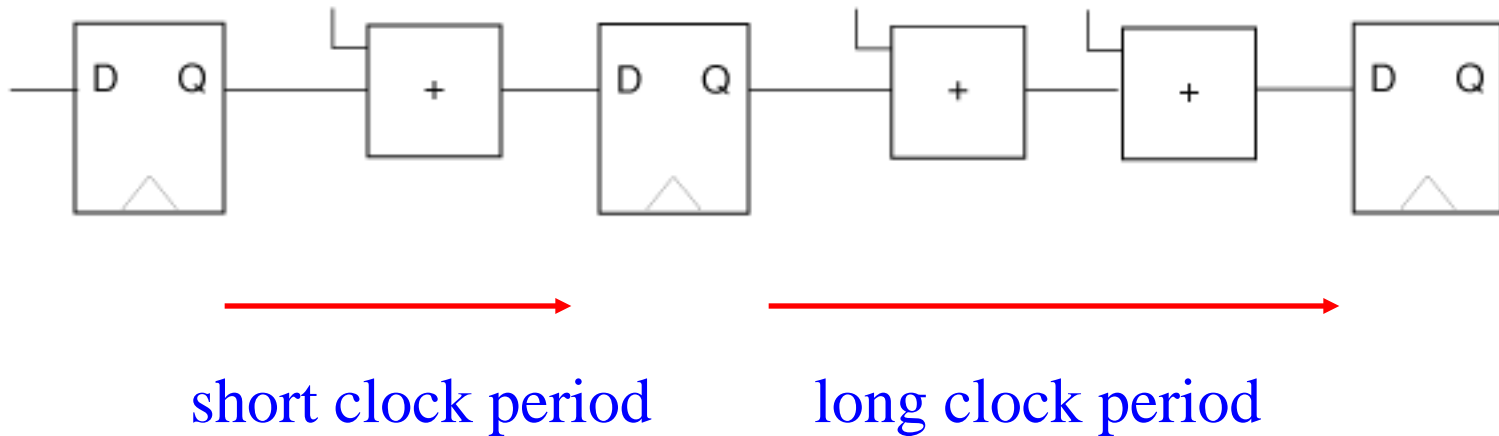
Qualified clocks and skew

- 
- Logic in the clocking path introduces delay.
 - Delay can cause clock to arrive at latches at different times, violating clocking assumptions.
 - When designing qualification logic:
 - minimize and check skew;
 - sharpen clock edge.

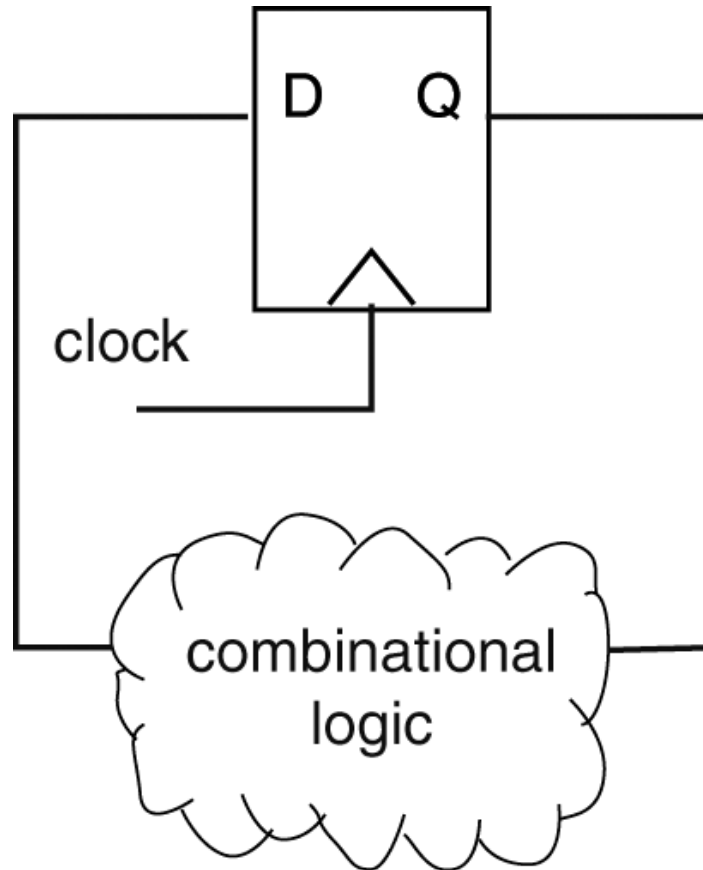
Unbalanced delays




Logic with unbalanced delays leads to inefficient use of logic:



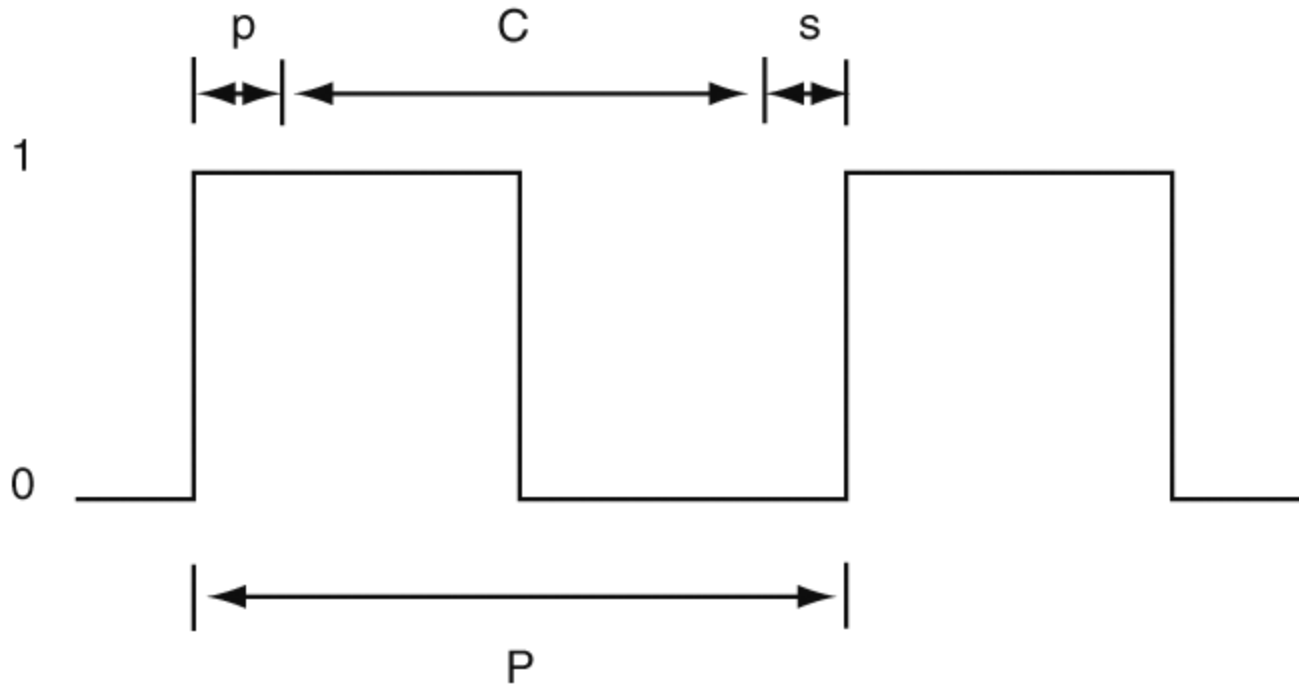
Flip-flop-based system performance analysis



Flip-flop-based system model

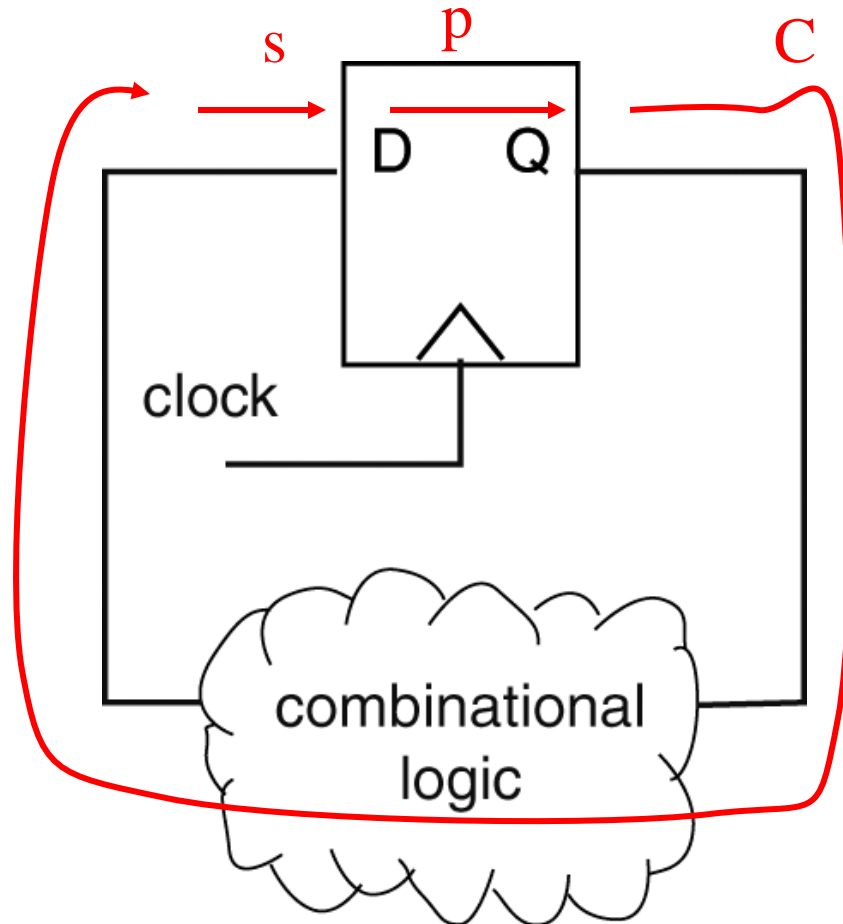
- 
- Clock signal is perfect (no rise/fall), period P .
 - Clock event on rising edge.
 - Setup time s .
 - Time from arrival of combinational logic event to clock event.
 - Propagation time p .
 - Time for value to go from flip-flop input to output.
 - Worst-case combinational delay C .
 - Time from output of flip-flop to input.

Clock parameters

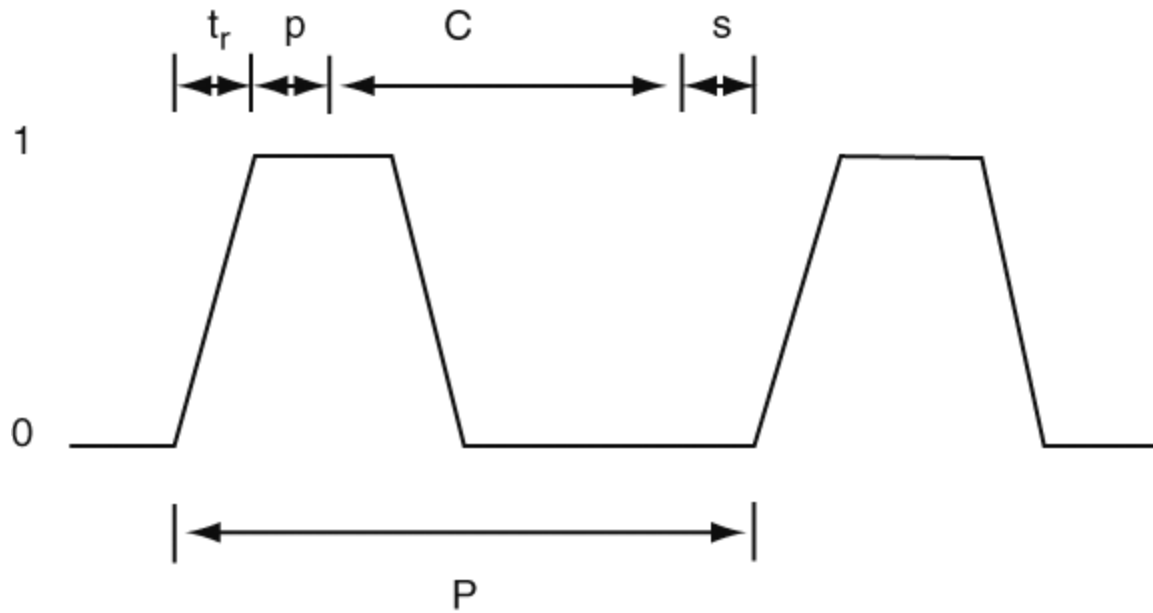


Clock period constraint

■ $P \geq C + s + p.$

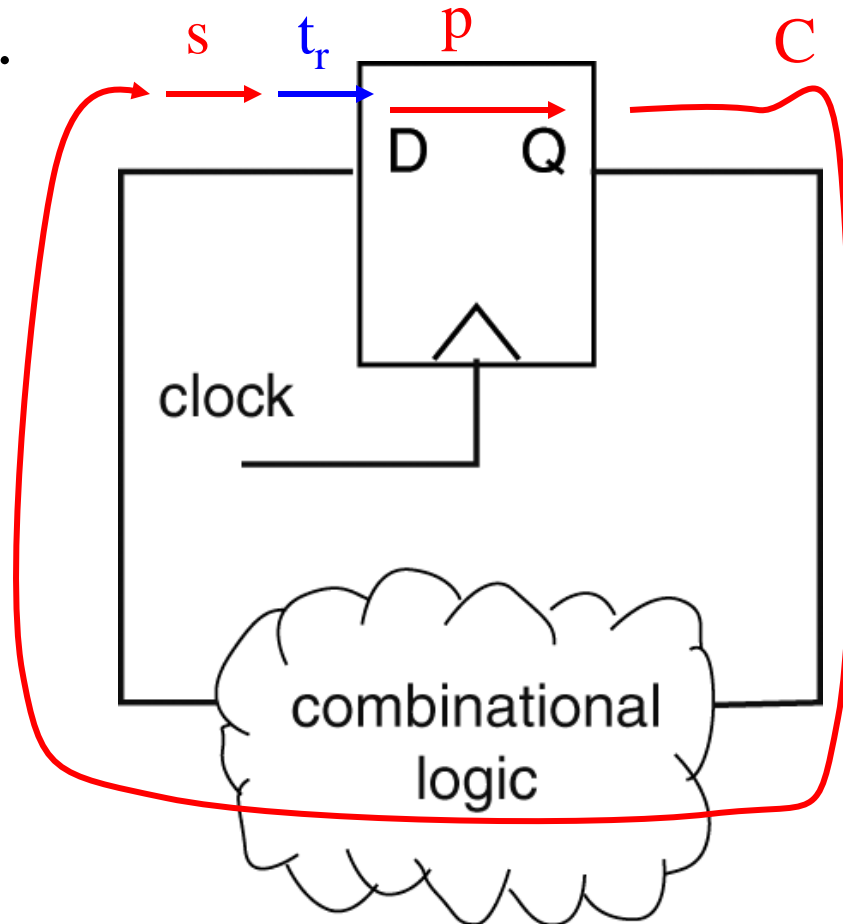


Clock with rise/fall



Rise/fall clock period constraint

■ $P \geq C + s + p + t_r$.

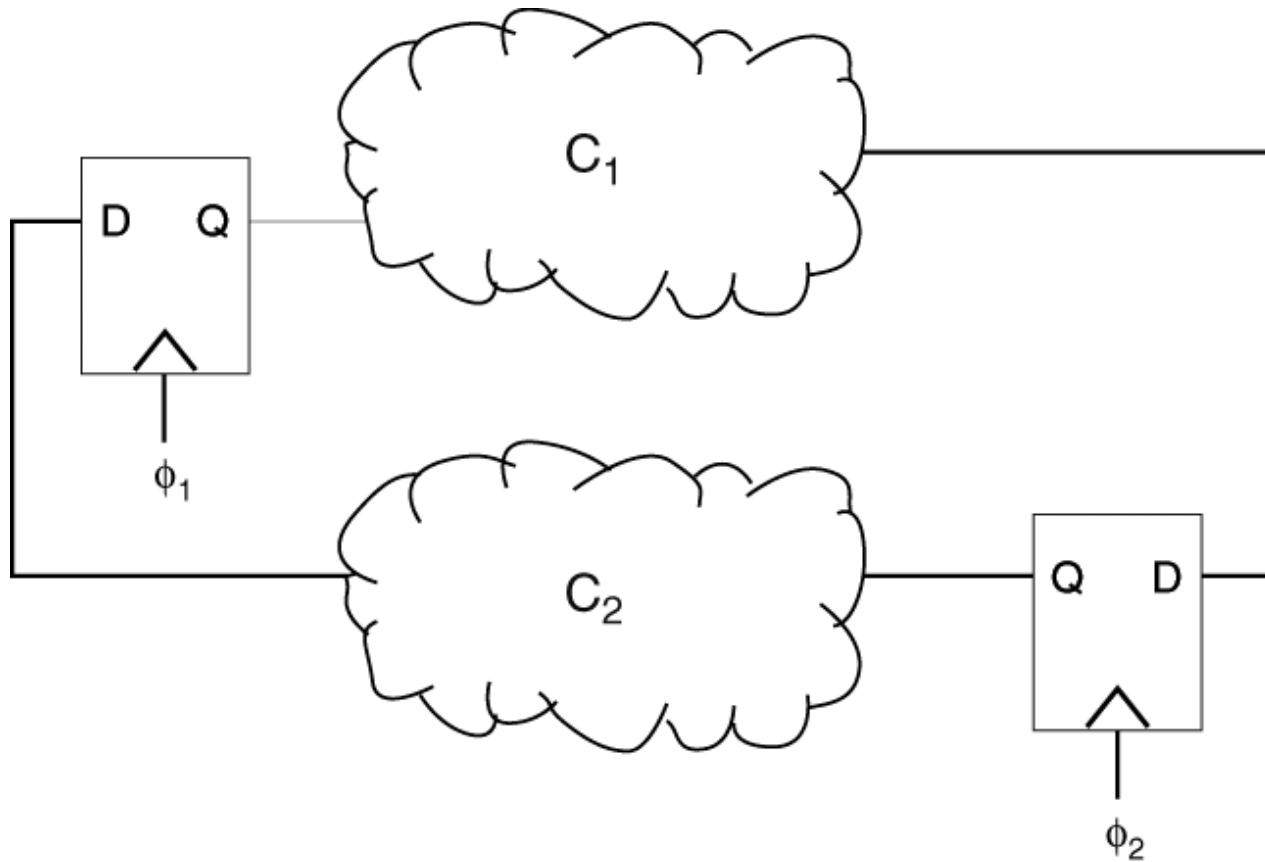


Latch system clock period

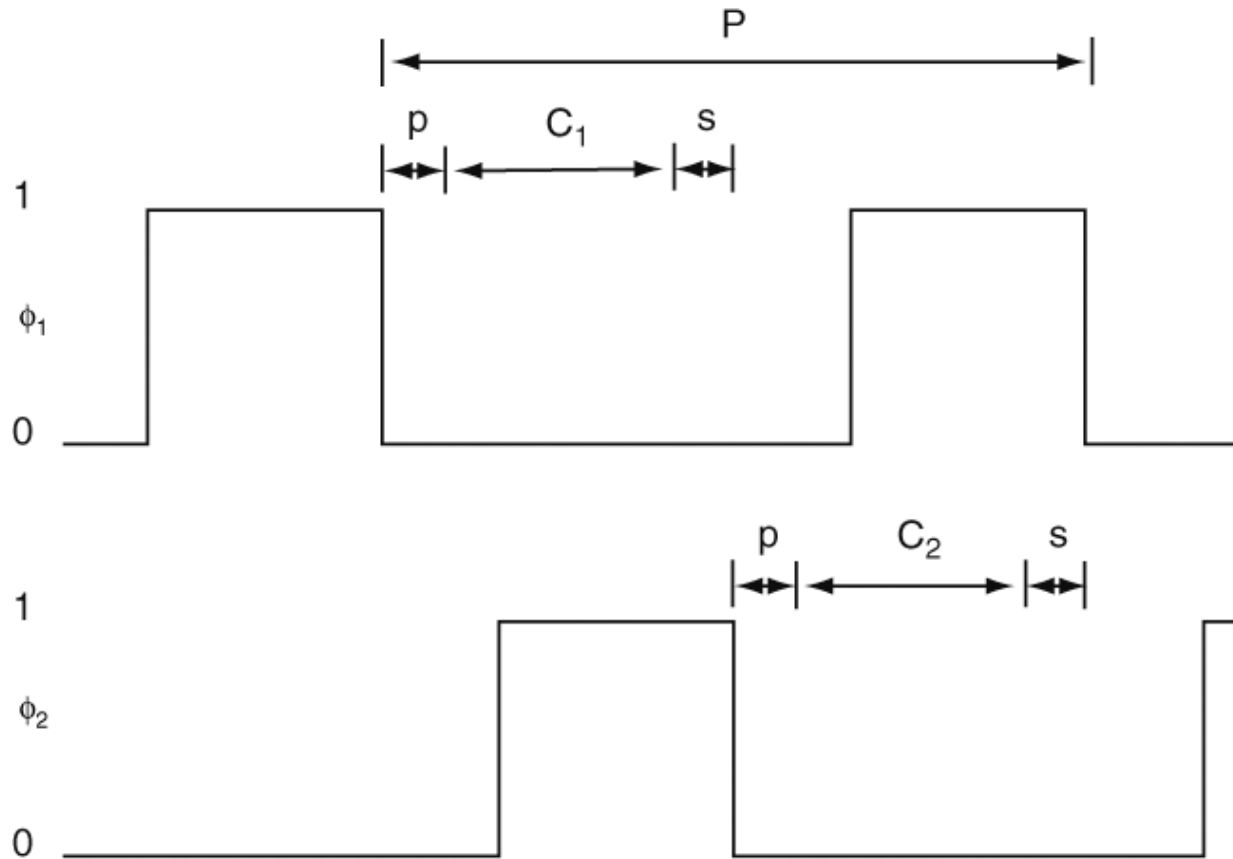


- For each phase, phase period must be longer than sum of:
 - combinational delay;
 - latch propagation delay.
- Phase period depends on longest path.


Latch-based system model



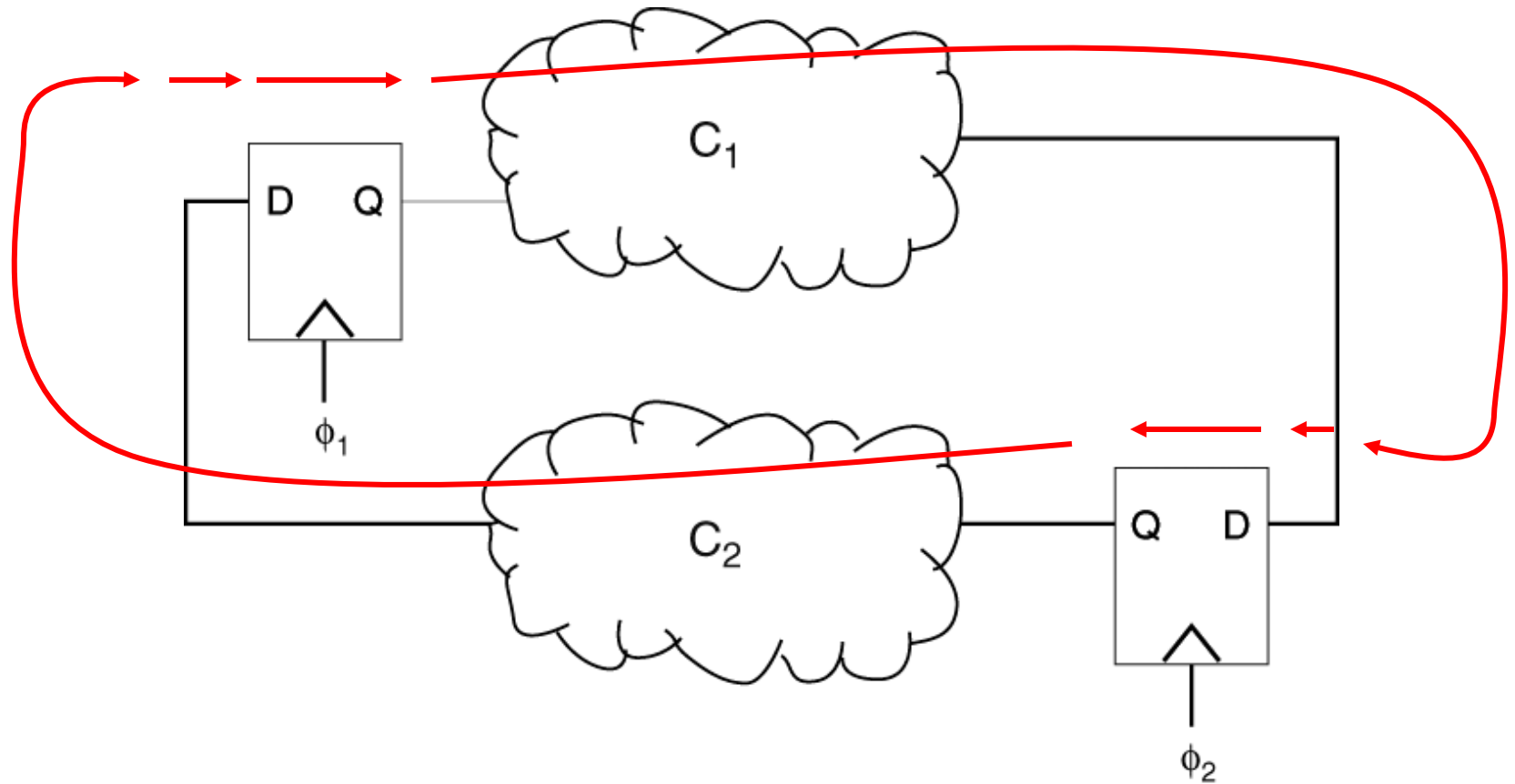
Two-phase timing parameters



Clock period constraint

- 
- Total clock period (both phases):
 - $P \geq C1 + C2 + 2s + 2p.$
 - Each phase must meet timing for its own latch.

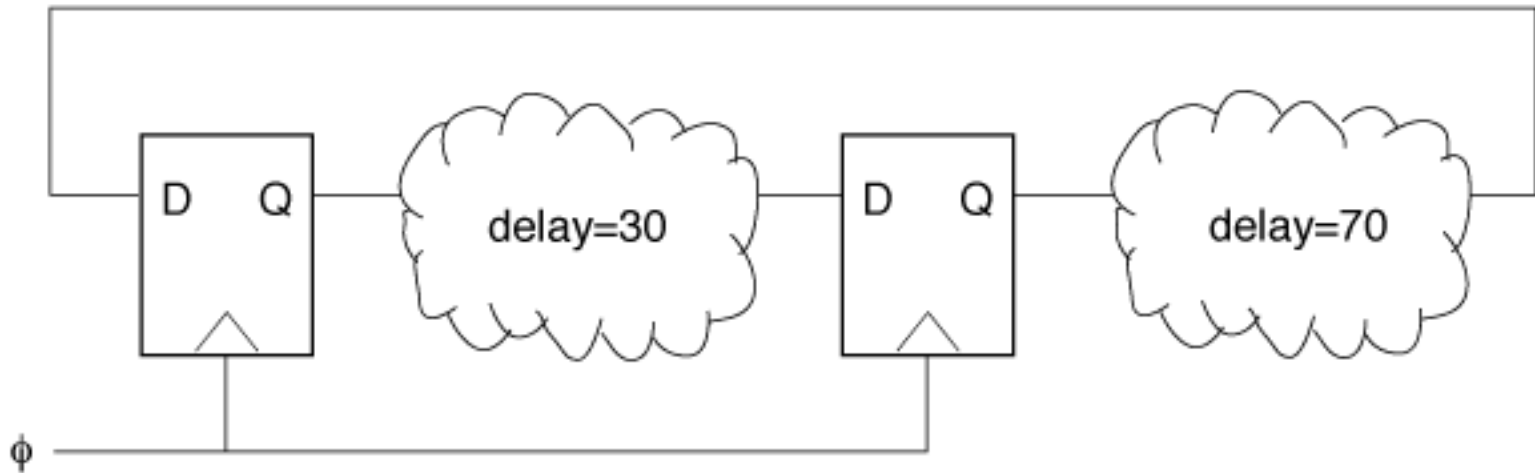
Latch-based system model



Example with unbalanced phases



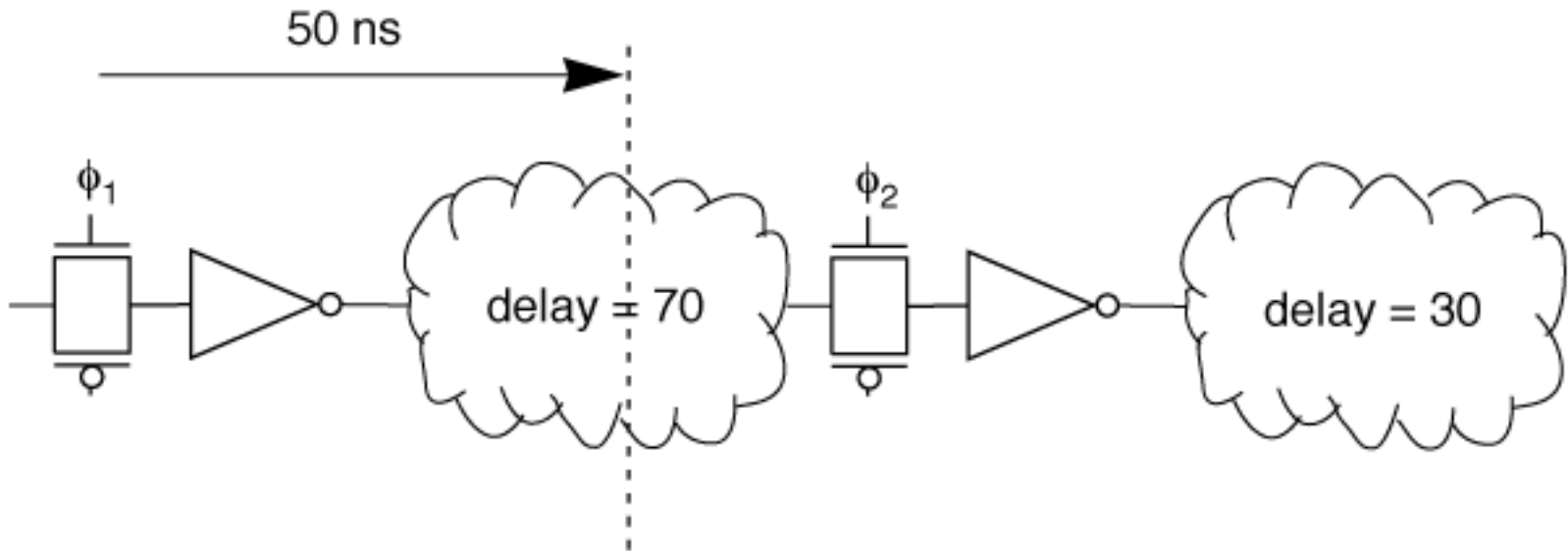
One phase is much longer than the other:



Spreading out a phase



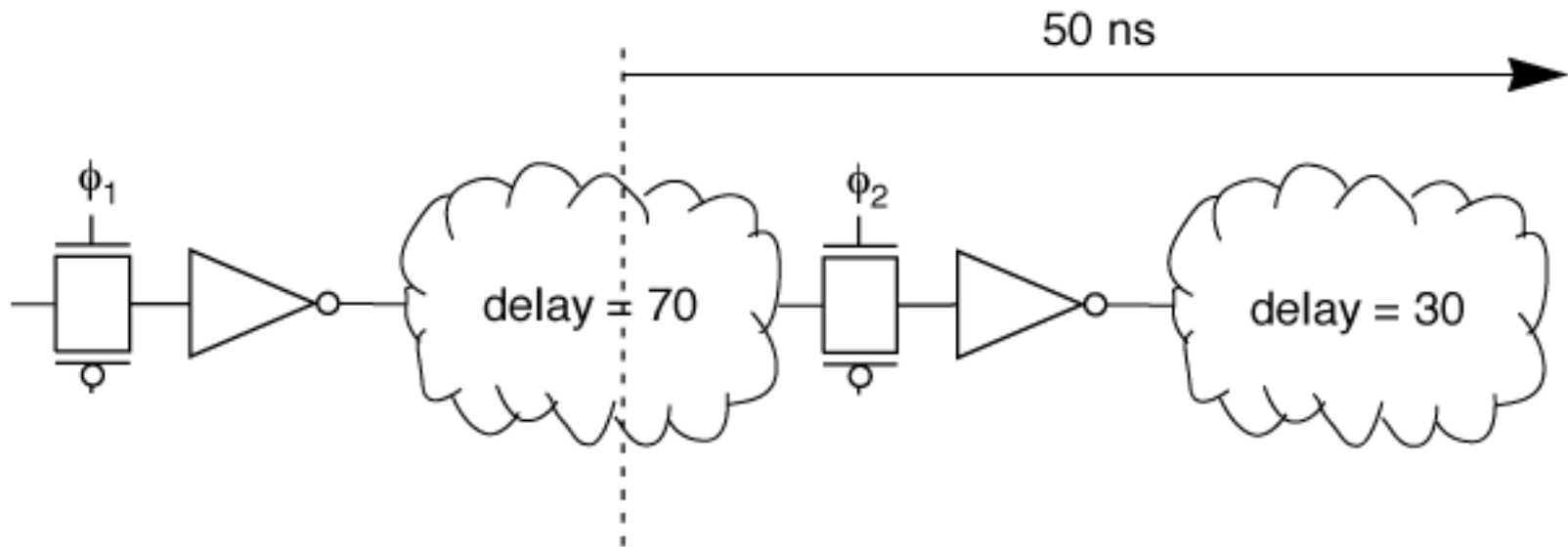
Compute only part of long paths in one phase:



Spreading out a phase, cont'd.



Use other phase for end of long logic block
and all of short logic block:



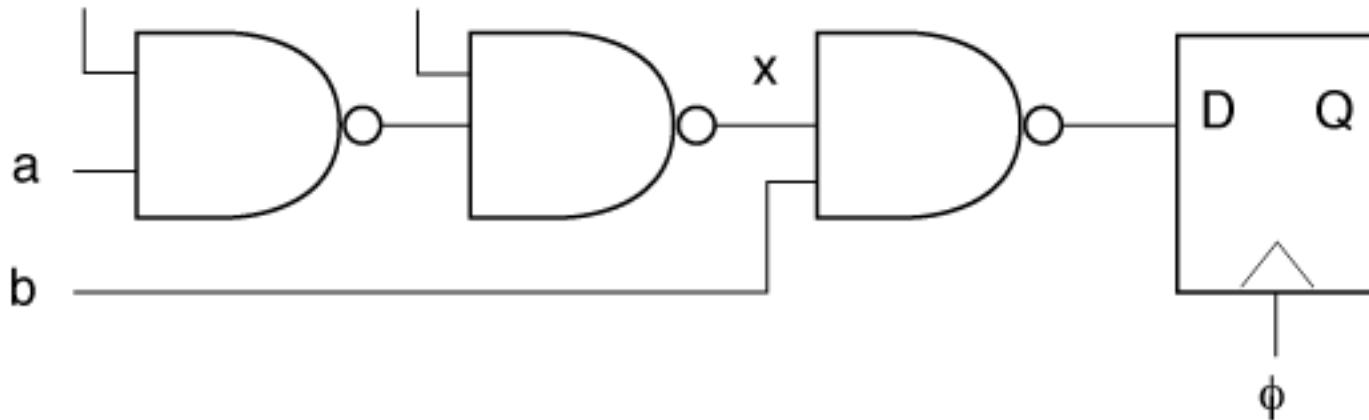
Skew



- **Skew**: relative delay between events.
- **Signal skew**: most important for asynchronous, timing-dependent logic.
- **Clock skew**: can harm any sequential system.

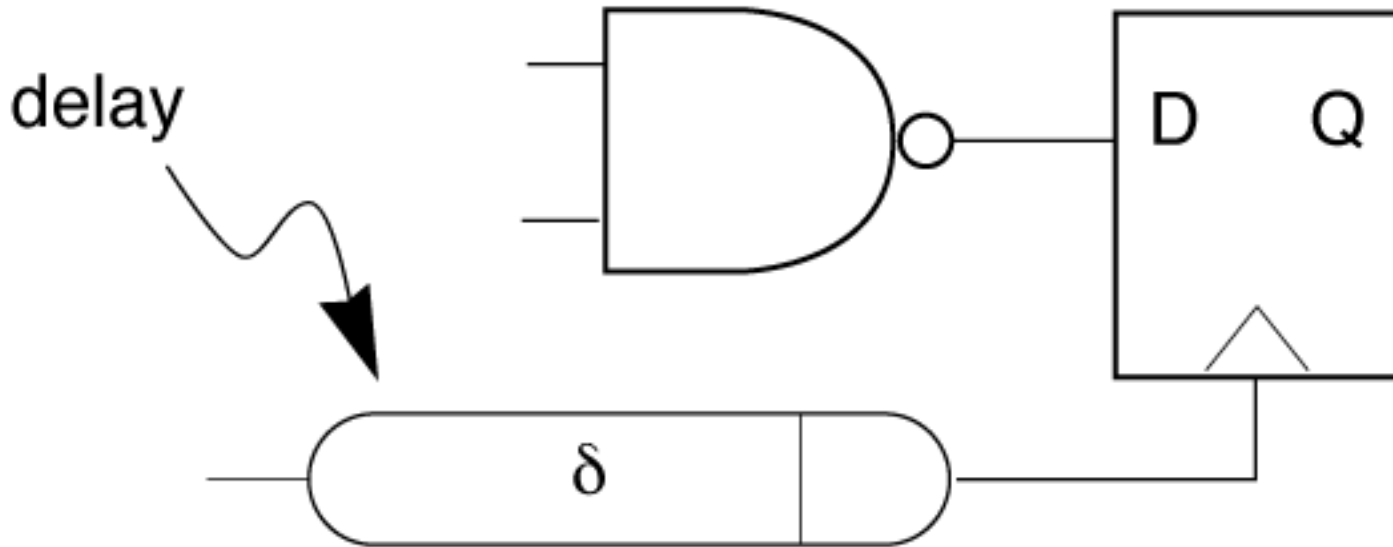
Signal skew

Machine data signals must obey setup and hold times—avoid signal skew.

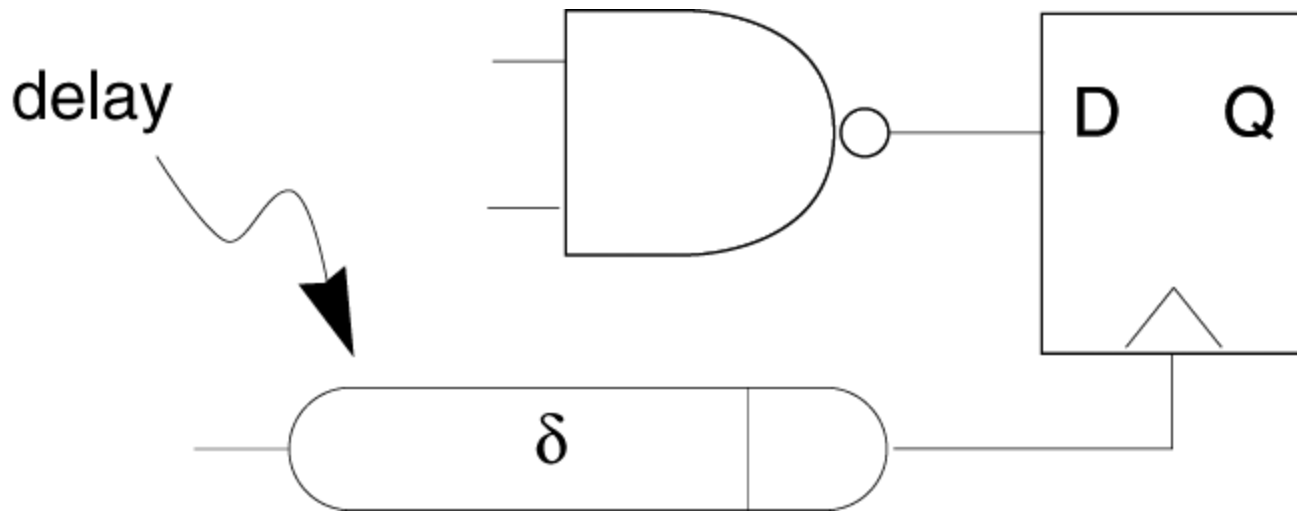


Clock skew

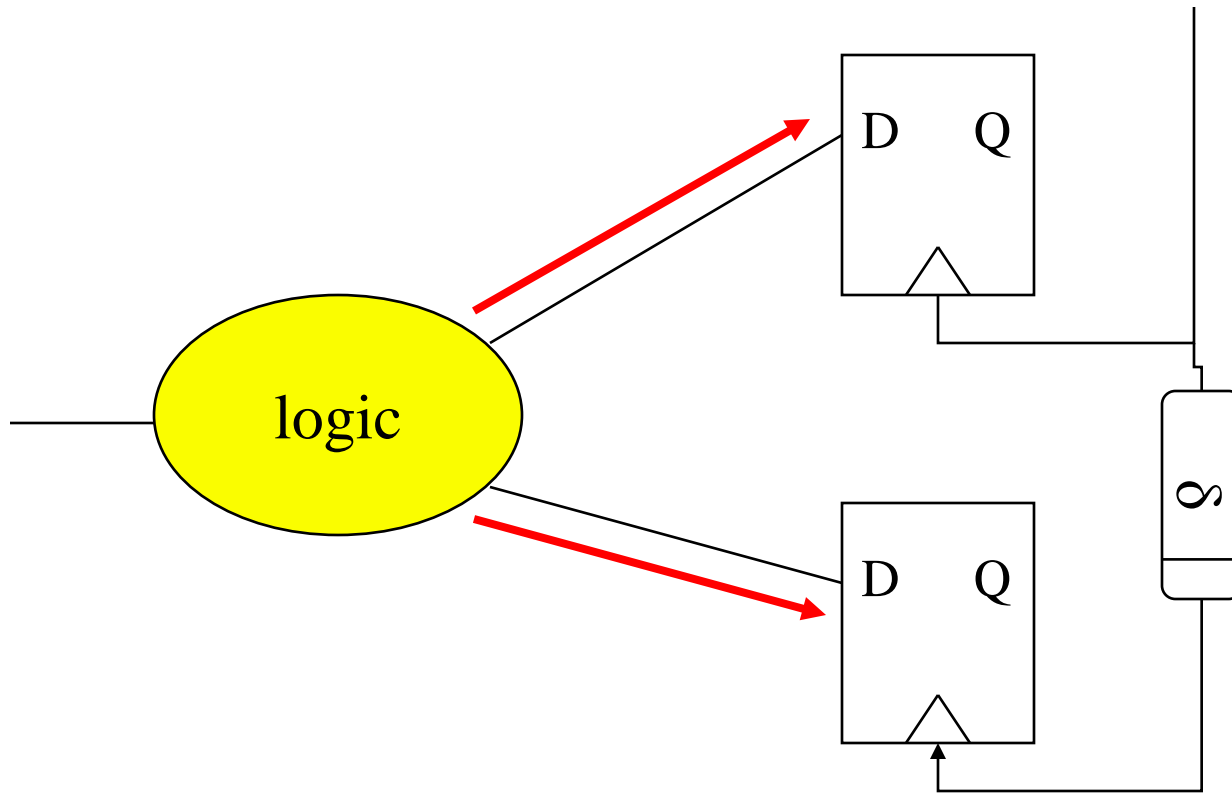
Clock must arrive at all memory elements in time to load data.



Clock skew example



Clock skew in system



Clock distribution

- Often one of the hardest problems in clock design.
 - Fast edges.
 - Minimum skew.

